Sampling Graphs with a Prescribed Joint Degree Distribution Using Markov Chains

Isabelle Stanton*
UC Berkeley
isabelle@eecs.berkeley.edu

Ali Pinar†
Sandia National Laboratories‡
apinar@sandia.gov

Abstract

One of the most influential results in network analysis is that many natural networks exhibit a power-law or log-normal degree distribution. This has inspired numerous generative models that match this property. However, more recent work has shown that while these generative models do have the right degree distribution, they are not good models for real life networks due to their differences on other important metrics like conductance. We believe this is, in part, because many of these real-world networks have very different *joint degree distributions*, i.e. the probability that a randomly selected edge will be between nodes of degree k and l. Assortativity is a sufficient statistic of the joint degree distribution, and it has been previously noted that social networks tend to be assortative, while biological and technological networks tend to be disassortative.

We suggest that the joint degree distribution of graphs is an interesting avenue of study for further research into network structure. We provide a simple greedy algorithm for constructing simple graphs from a given joint degree distribution, and a Monte Carlo Markov Chain method for sampling them. We also show that the state space of simple graphs with a fixed degree distribution is connected via *endpoint switches*. We empirically evaluate the mixing time of this Markov Chain by using experiments based on the autocorrelation of each edge.

*Supported by a National Defense Science and Engineering Graduate Fellowship. Work performed while at Sandia National Laboratories, Livermore CA.

†This work is supported by the DOE ASCR Applied Mathematics Program.

‡Sandia National Laboratories is a multi-program laboratory operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energys National Nuclear Security Administration under contract DE-AC04-94AL85000.

1 Introduction

Graphs are widely recognized as the standard modeling language for many complex systems, including physical infrastructure (e.g., Internet, electric power, water, and gas networks), scientific processes (e.g., chemical kinetics, protein interactions, and regulatory networks in biology starting at the gene levels, all the way up to ecological systems), and relational networks (e.g., citation networks, hyperlinks on the web, and social networks). The broader adoption of the graph models over the last decade, along with the growing importance of associated applications, calls for descriptive and generative models for real networks. What is common among these networks? How do they differ statistically? Can we quantify the differences among these networks? Answering these questions requires understanding the topological properties of these graphs, which have lead to numerous studies on topological properties of many "real-world" networks from the Internet to social, biological and technological networks [6].

Perhaps the most prominent result coming out of these studies is the existence of power-law or log-normal distributions over many quantities and in particular the degree distribution: the number of nodes of degree k is proportional to $k^{-\alpha}$. The ubiquity of this distribution has been a motivator for many different generative models, like preferential attachment, the copying model, the Barabasi hierarchical model, forest-fire model, the Kronecker graph model and geometric preferential attachment [7, 16, 18, 29, 17]. Many of these models also match other observed features, such as small diameter or densification [14]. However, recent studies comparing the generative models with real networks on metrics like conductance show that the models do not match other important features of the networks [19].

The degree distribution alone does not define a graph. McKay's estimate shows that there may be exponentially many graphs with the same degree distribution. However, models based on degree distribution are commonly used to compute statistically signif-

icant structures in a graph. For example, the modularity metric is a standard metric to find communities in graphs [24, 23]. This metric defines a null hypothesis for the structure of a graph based on its degree distribution, namely that probability of an edge between vertex v_i and v_j is proportional to $d_i d_j$, where d_i and d_j represent the degrees of vertices v_i and v_j. The modularity of a group of vertices is defined by how much their structure deviates from the null hypothesis, and a higher modularity signifies a better community. The key point here is that the null hypothesis is solely based on its degree distribution and therefore might be incorrect. As a result, better descriptive models are critically important.

One way to enhance the results based on degree distribution is to use a more restrictive feature such as the joint degree distribution. Intuitively, if degree distribution of a graph describes the probability that a vertex selected uniformly at random will be of degree k then its joint degree distribution describes the probability that a randomly selected *edge* will be between nodes of degree k and l. Note that while the joint degree distribution uniquely defines the degree distribution of a graph up to isolated nodes, graphs with the same degree distribution may have very different joint degree distributions. For example, the assortativity of a network measures whether nodes prefer to attach to other similar or dissimilar nodes. When similarity is defined in terms of a node's degree, it is a sufficient statistic of the joint degree distribution and measures how different the joint degree distribution is from one where all of the edges are between nodes of the same degree. Studies of the assortativity of networks show that social networks tend to be assortative, while biological and technological networks like the internet tend to be dissortative [26, 25].

Before attempting to use the joint degree distribution as a metric for designing generative models, it is important to know how tractable it is to work with. The primary questions investigated by this paper are: Given a joint degree distribution and an integer n, is it possible to construct a graph of size n with that joint degree distribution? Is it possible to construct or generate a *uniformly random* graph with that same joint degree distribution? We address both of these problems in this paper both from a theoretical and from an empirical perspective.

Contributions. We make several contributions to this problem, both theoretically and experimentally. First, we discuss the necessary and sufficient conditions for a given joint degree vector to be graphical. We prove that these conditions are sufficient by providing a new constructive algorithm. Next, we introduce a new configuration model for the joint degree matrix problem which is a natural extension of the configuration model for the degree sequence problem. Finally, using this configuration model, we develop Markov Chains for sampling both pseudographs and simple graphs with a fixed joint degree matrix. We prove the correctness of both chains and mixing time for the pseudograph chain by using previous work. The mixing time of the simple graph chain is experimentally evaluated using autocorrelation.

In practice, Monte Carlo Markov Chains are a very popular method for sampling from difficult distributions. However, it is often very difficult to theoretically evaluate the mixing time of the chain, and many practitioners simply stop the chain after 5,000, 10,000 or 20,000 iterations without much justification. Our experimental design with autocorrelation provides a set of statistics that can be used as a justification for choosing a stopping point.

Related work. The related work can be roughly divided into two categories: constructing and sampling graphs with a fixed degree distribution using sequential importance sampling or Monte Carlo Markov Chain methods, and experimental work on heuristics for generating random graphs with a fixed joint degree distribution.

The methods for constructing graphs with a given degree distribution are primarily either reductions to perfect matchings or sequential sampling methods. There are two popular perfect matching methods. The first is the *configuration model* [1]: k mini-vertices are created for each degree k vertex, and all the mini-vertices are connected. Given any perfect matching in the configuration the edges in the graph correspond to the connected mini-vertices. This allows multiple edges and self-loops, which are often undesirable. The second approach, the *gadget configuration model*, prevents this problem by creating a gadget for each vertex. If v_i has degree d_i, then it is replaced with a complete bipartite graph (U_i, V_i) with $|U_i| = n - 1 - d_i$ and $|V_i| = n - 1$. Exactly one node in each V_i is connected to each other V_j, representing edge (i, j) [12]. Any perfect matching in this model corresponds exactly to a simple graph. These models are pictured in Figures 1 and 2 respectively in the Appendix. We use a natural extension of the first configuration model to the joint degree distribution problem.

There are also sequential sampling methods that will construct a graph with a given degree distribution. Some of these are based on the necessary and sufficient Erdős-Gallai conditions for a degree sequence to be graphical [4], while others follow the method of Steger and Wormald [3, 32, 30, 10, 13]. These combine the construction and sampling parts of the problem and can

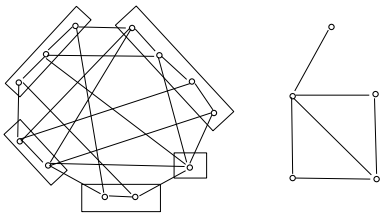

Figure 1: On the left, we see an example of the configuration model of the graph on the right. Each vertex is split into a number of minivertices equal to it's degree, and then all minivertices are connected. Not all edges are shown for clarity.

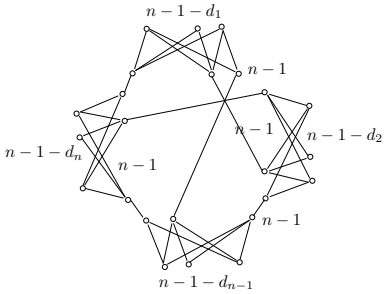

Figure 2: The gadget configuration model. A gadget is created for each vertex and there are 4 shown above. One half of the gadget is $n-1$ vertices, and the other half is $n-1-d_i$, where d_i is the degree. Then each gadget is connected once to each other gadget. A perfect matching in this graph corresponds to a graph with the correct degree sequence.

be quite fast. The current best work can sample graphs where $d_{max} = O(m^{1/4-\tau})$ in $O(md_{max})$ time [3].

Another approach for sampling graphs with a given degree distribution is to use a Monte Carlo Markov Chain method. There is significant work on sampling perfect matchings [11, 5]. There has also been work specifically targeted at the degree distribution problem. Kannan, Tetali and Vempala [12] analyze the mixing time of a Markov Chain that mixes on the configuration model, and another for the gadget configuration model. Gkantsidis, Mihail and Zegura [9] use a Markov Chain on the configuration model, but reject any transition that creates a self-loop, multiple edge or disconnects the graph. Both of these chains use the work of Taylor [33] to argue that the state space is connected.

Amanatidis, Green and Mihail study the problem of when a joint degree matrix has graphical representation and when a connected representation exists [2]. They give necessary and sufficient conditions for both of these problems, and constructive algorithms. In Section 2, we give a simpler constructive algorithm for creating a graphical representation that is based on solving the degree sequence problem instead of alternating structures.

Another vein of related work is that of Mahadevan et al. who introduce the concept of dK-series [22, 21]. In this model, d refers to the dimension of the distribution and $2K$ is the joint degree distribution. They propose a heuristic for generating random $2K$-graphs for a fixed $2K$ distribution via edge rewirings. However, their method can get stuck if there is only 1 node with any degree k and the state space is not connected. We provide a theoretically sound method of doing this.

Finally, Newman also studies the problem of fixing an assortativity value, finding a *joint remaining degree distribution* with that value, and then sampling a random graph with that distribution using Markov Chains [26, 25]. His Markov Chain starts at any graph with the correct degree distribution and converges to a pseudograph with the correct joint remaining degree distribution. By contrast, our work provides a theoretically sound way of constructing a simple graph with a given joint degree distribution first, and our Markov Chain only has simple graphs with the same joint degree distribution as its state space.

Notation and Definitions Formally, a degree distribution of a graph is the probability that a node chosen at random will be of degree k. Similarly, the joint degree distribution is the probability that a randomly selected *edge* will have endpoints of degree k and l. In this paper, we are concerned with constructing graphs that exactly match these distributions, so rather than probabilities, we will use a counting definition below and call it the *joint degree matrix*. In particular, we will be concerned with generating *simple* graphs that do not contain multiple edges or self-loops.

Definition 1. *The degree vector (DV) $d(G)$ of a graph G is a vector where $d(G)_k$ is the number of nodes of degree k in G.*

A generic degree vector will be denoted by $\mathcal{D}$.

Definition 2. *The joint degree matrix (JDM) $\mathcal{J}(G)$ of a graph G is a matrix where $\mathcal{J}(G)_{k,l}$ is exactly the number of edges between nodes of degree k and degree l in G.*

A generic joint degree matrix will be denoted by $\mathcal{J}$. Given a joint degree matrix, $\mathcal{J}$, we can recover the number of edges in the graph as $m = \sum_{k=1}^{\infty}\sum_{l=k}^{\infty}\mathcal{J}_{k,l}$. We can also recover the degree vector as $\mathcal{D}_k = \frac{1}{k}(\mathcal{J}_{k,k} + \sum_{l=1}^{\infty}\mathcal{J}_{k,l})$. The term $\mathcal{J}_{k,k}$ is added twice because $k\mathcal{D}_k$ is the number of endpoints of degree k and the edges in $\mathcal{J}_{k,k}$ contribute two endpoints.

The number of nodes, n is then $\sum_{k=1}^{\infty} \mathcal{D}_k$. This count does not include any degree 0 vertices, as these have no edges in the joint degree matrix. Given n and m, we can easily get the degree distribution and joint degree distribution. They are $P(k) = \frac{1}{n}\mathcal{D}_k$ while $P(k,l) = \frac{1}{m}\mathcal{J}_{k,l}$. Note that $P(k)$ is not quite the marginal of $P(k,l)$ although it is closely related.

The Joint Degree Matrix Configuration Model We propose a new configuration model for the joint degree distribution problem. Given $\mathcal{J}$ and its corresponding $\mathcal{D}$ we create k mini-vertices for every vertex of degree k. In addition, for every edge with endpoints of degree k and l we create two mini-endpoints, one of class k and one of class l. We connect all k degree mini-vertices to the class k mini-endpoints. This forms a complete bipartite graph for each degree, and each of these forms a disconnected component. We will call each of these components the "k-neighborhood". Notice that there are $k\mathcal{D}_k$ mini-vertices of degree k, and $k\mathcal{D}_k = \mathcal{J}_{k,k} + \sum_l \mathcal{J}_{k,l}$ corresponding mini-endpoints in each k-neighborhood. This is pictured in Figure 3 in the Appendix.

Take any perfect matching in this graph. If we merge each pair of mini-endpoints that correspond to the same edge, we will have some pseudograph that has exactly the desired joint degree matrix. This observation forms the basis of our sampling method.

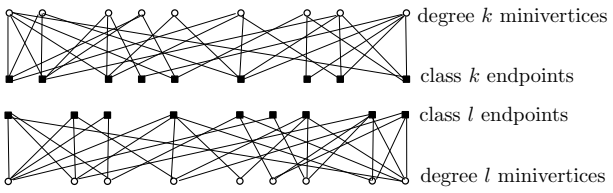

Figure 3: The joint degree matrix configuration model. This shows just two degree neighborhoods of the joint degree matrix configuration model. Each vertex of degree k is split into k minivertices which are represented by the circles. These then form a complete bipartite component when they are connected with the class k endpoints, the squares. Each degree neighborhood is completely disconnected from all others.

2 Constructing Graphs with a Given Joint Degree Matrix

The Erdős-Gallai condition is a necessary and sufficient condition for a degree sequence to be realizable as a simple graph.

THEOREM 2.1. **Erdős-Gallai** *A degree sequence $\overline{d} = \{d_1, d_2, \cdots d_n\}$ sorted in non-increasing order is graphical if and only if for every $k \leq n$, $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min(d_i, k)$.*

The necessity of this condition comes from noting that in a set of vertices of size k, there can be at most $\binom{k}{2}$ internal edges, and for each vertex v not in the subset, there can be at most $\min\{d(v), k\}$ edges entering. The condition considers each subset of decreasing degree vertices and looks at the degree requirements of those nodes. If the requirement is more than the available edges, the sequence can not be graphical. The sufficiency is shown via the constructive Havel-Hakimi algorithm.

The existence of the Erdős-Gallai condition inspires us to ask whether similar necessary and sufficient conditions exist for a joint degree matrix to be graphical. The following necessary and sufficient conditions are due to Amanatidis et al. [2].

THEOREM 2.2. *Let $\mathcal{J}$ be given and $\mathcal{D}$ be the associated degree distribution. $\mathcal{J}$ can be realized as a simple graph if and only if (1) $\mathcal{D}_k$ is integer-valued for all k and (2) $\forall k, l$, if $k \neq l$ then $\mathcal{J}_{k,l} \leq \mathcal{D}_k\mathcal{D}_l$. Otherwise, $\forall k$ $\mathcal{J}_{k,k} \leq \binom{\mathcal{D}_k}{2}$.*

The necessity of these conditions is clear. The first condition requires that there are an integer number of nodes of each degree value. The next two are that the number of edges between nodes of degree k and l (or k and k) are not more than the total possible number of k to l edges in a simple graph defined by the marginal degree sequences. Amanatidis et al. show the sufficiency through a constructive algorithm. We will now introduce a new algorithm that runs in $O(md_{max})$ time.

The algorithm proceeds by building a nearly regular graph for each class of edges, $\mathcal{J}_{k,l}$. Assume that $k \neq l$ for simplicity. Each of the $\mathcal{D}_k$ nodes of degree k receives $\lfloor \mathcal{J}_{k,l}/\mathcal{D}_k \rfloor$ edges, while $\mathcal{J}_{k,l} \mod \mathcal{D}_k$ each have an extra edge. Similarly, the l degree nodes have $\lfloor \mathcal{J}_{k,l}/\mathcal{D}_l \rfloor$ edges, with $\mathcal{J}_{k,l} \mod \mathcal{D}_l$ having 1 extra. We can then construct a simple bipartite graph with this degree sequence. This can be done in linear time in the number of edges using queues as is discussed after Observation 2.1. If $k = l$, the only differences are that the graph is no longer bipartite and there are $2\mathcal{J}_{k,k}$ endpoints to be distributed among $\mathcal{D}_k$ nodes. To find a simple nearly regular graph, one can use Bayati, Kim and Saberi's [3] algorithm in $O(\mathcal{J}_{k,k}k)$ time.

We must show that there is a way to combine all of these nearly-regular graphs together without violating any degree constraints. Let $d = \langle d_1, d_2, \cdots d_n \rangle$ be the sorted nonincreasing order degree sequence from $\mathcal{D}$. Let $\hat{d}_v$ denote the residual degree sequence where the residual degree of a vertex v is d_v minus the number of edges that currently neighbor v. Also, let $\hat{\mathcal{D}}_k$ denote the

number of nodes of degree k that have non-zero residual degree, i.e. $\hat{\mathcal{D}}_k = \sum_{d_j=k} \mathbf{1}(\hat{d}_j \neq 0)$.

ALGORITHM 2.1.
1: **for** $k = n \cdots 1$ **and** $l = k \cdots 1$ **do**
2: **if** $k \neq l$ **then**
3: Let $a = \mathcal{J}_{k,l} \mod \mathcal{D}_k$ and $b = \mathcal{J}_{k,l} \mod \mathcal{D}_l$
4: Let $x_1 \cdots x_a = \lfloor \frac{\mathcal{J}_{k,l}}{\mathcal{D}_k} \rfloor + 1$, $x_{a+1} \cdots x_{\mathcal{D}_k} = \lfloor \frac{\mathcal{J}_{k,l}}{\mathcal{D}_k} \rfloor$ and $y_1 \cdots y_b = \lfloor \frac{\mathcal{J}_{k,l}}{\mathcal{D}_l} \rfloor + 1$, $y_{b+1} \cdots y_{\mathcal{D}_l} = \lfloor \frac{\mathcal{J}_{k,l}}{\mathcal{D}_l} \rfloor$
5: Construct a simple bipartite graph B with degree sequence $x_1 \cdots x_{\mathcal{D}_k}, y_1 \cdots y_{\mathcal{D}_l}$
6: **else**
7: Let $c = 2\mathcal{J}_{k,k} \mod \mathcal{D}_k$
8: Let $x_1 \cdots x_c = \lfloor \frac{2\mathcal{J}_{k,k}}{\mathcal{D}_k} \rfloor + 1$ and $x_{c+1} \cdots x_{\mathcal{D}_k} = \lfloor \frac{2\mathcal{J}_{k,k}}{\mathcal{D}_k} \rfloor$
9: Construct a simple graph B with the degree sequence $x_1 \cdots x_{\mathcal{D}_k}$
10: **end if**
11: Place B into G by matching the nodes of degree k with higher residual degree with $x_1 \cdots x_a$ and those of degree l with higher residual degree with $y_1 \cdots y_b$. The other vertices in B can be matched in any way with those in G of degree k and l
12: Update the residual degrees of each k and l degree node.
13: **end for**

To combine the nearly uniform subgraphs, we start with the largest degree nodes, and the corresponding largest degree classes. First, we note that after every iteration, the joint degree sequence is still feasible if $\forall k, l, k \neq l$ $\hat{\mathcal{J}}_{k,l} \leq \hat{\mathcal{D}}_k \hat{\mathcal{D}}_l$ and $\forall k$ $\hat{\mathcal{J}}_{k,k} \leq \binom{\hat{\mathcal{D}}_k}{2}$.

We will prove that Algorithm 2.1 can always satisfy the feasibility conditions. First, we note a fact.

OBSERVATION 1. *For all k, $\sum_l \hat{\mathcal{J}}_{k,l} + \hat{\mathcal{J}}_{k,k} = \sum_{d_j=k} \hat{d}_j$*

This follows directly from the fact that the left hand side is summing over all of the k endpoints needed by $\hat{\mathcal{J}}$ while the right hand side is summing up the available residual endpoints from the degree distribution. Next, we note that if all residual degrees for degree k nodes are either 0 or 1, then:

OBSERVATION 2. *If, for all j such that $d_j = k$, $\hat{d}_j = 0$ or 1 then $\sum_{d_j=k} \hat{d}_j = \sum_{d_j=k} \mathbf{1}(\hat{d}_j \neq 0) = \hat{\mathcal{D}}_k$.*

LEMMA 2.1. *After every iteration, for every pair of vertices u, v of any degree k, $|\hat{d}_u - \hat{d}_v| \leq 1$.*

Amanatidis et al. refer to Lemma 2.1 as the *balanced degree invariant*. This is most easily proven by considering the vertices of degree k as a queue. If there are x edges to be assigned, we can consider the process of deciding how many edges to assign each vertex as being one of popping vertices from the top of the queue and reinserting them at the end x times. Each vertex is assigned edges equal to the number of times it was popped. The next time we assign edges with endpoints of degree k, we start with the queue at the same position as where we ended previously. It is clear that no vertex can be popped twice without all other vertices being popped at least once.

LEMMA 2.2. *The above algorithm can always greedily produce a graph that satisfies $\mathcal{J}$, provided $\mathcal{J}$ satisfies the initial necessary conditions.*

Proof. There is one key observation about this algorithm - it maximizes $\hat{\mathcal{D}}_k \hat{\mathcal{D}}_l$ by ensuring that the residual degrees of any two vertices of the same degree never differ by more than 1. By maximizing the number of available vertices, we can not get stuck adding a self-loop or multiple edge. From this, we gather that if, for some degree k, there exists a vertex j such that $\hat{d}_j = 0$, then for all vertices of degree k, their residuals must be either 0 or 1. This means that $\sum_{d_j=k} \hat{d}_j = \hat{\mathcal{D}}_k \geq \hat{\mathcal{J}}_{k,l}$ for every other l from Observation 2.

From the initial conditions, we have that for every k, l $\mathcal{J}_{k,l} \leq \mathcal{D}_k \mathcal{D}_l$. $\mathcal{D}_k = \hat{\mathcal{D}}_k$ provided that all degree k vertices have non-zero residuals. Otherwise, for any unprocessed pair, $\mathcal{J}_{k,l} \leq \min\{\hat{\mathcal{D}}_k, \hat{\mathcal{D}}_l\} \leq \hat{\mathcal{D}}_k \hat{\mathcal{D}}_l$. For the k, k case, it is clear that $\mathcal{J}_{k,k} \leq \hat{\mathcal{D}}_k \leq \binom{\hat{\mathcal{D}}_k}{2}$. Therefore, the residual joint degree matrix and degree sequence will always be feasible, and the algorithm can always continue.

THEOREM 2.3. *The necessary conditions for a joint degree matrix to be graphical imply that the associated degree vector satisfies the Erdős-Gallai condition.*

The proof is included in the Appendix.

3 Uniformly Sampling Graphs with Monte Carlo Markov Chain (MCMC) Methods

We now turn our attention to uniformly sampling graphs with a given graphical joint degree matrix using MCMC methods. We return to the joint degree matrix configuration model. We can obtain a starting configuration for any graphical joint degree matrix by using Algorithm 2.1. The transitions we use select any endpoint uniformly at random, then select any other endpoint in its degree neighborhood and swap the two edges that these neighbor. A more complex version of this chain checks that this swap does not create a multiple edge or self-loop. Formally, the transition function is a randomized algorithm given by Algorithm 3.1.

ALGORITHM 3.1. 1: With probability 0.5, stay at configuration C. Else:
2: Select any endpoint e_1 uniformly at random. It neighbors a vertex v_1 in configuration C
3: Select any e_2 u.a.r from e_1's degree neighborhood. It neighbors v_2
4: (Optional: If the graph obtained from the configuration with edges $E \cup \{(e_1, v_2), (e_2, v_1)\} \setminus \{(e_1, v_1), (e_2, v_2)\}$ contains a multi-edge or self-loop, reject)
5: $E \leftarrow E \cup \{(e_1, v_2), (e_2, v_1)\} \setminus \{(e_1, v_1), (e_2, v_2)\}$

There are two chains described by Algorithm 3.1. The first, $\mathcal{A}$ doesn't have step (4) and its state space is all pseudographs with the desired joint degree matrix. The second, $\mathcal{B}$ includes step (4) and only considers simple graphs with the right joint degree matrix.

We remind the reader of the standard result that any irreducible, aperiodic Markov Chain with symmetric transitions converges to the uniform distribution over its state space. Both $\mathcal{A}$ and $\mathcal{B}$ are aperiodic, due to the self-loop to each state. From the description of the transition function, we can see that $\mathcal{A}$ is symmetric. This is less clear for the transition function of $\mathcal{B}$. Is it possible for two connected configurations to have a different number of feasible transitions in a given degree neighborhood? We show that it is not the case in the following lemma. The proof is included in the appendix.

LEMMA 3.1. *The transition function of $\mathcal{B}$ is symmetric.*

The remaining important question is the connectivity of the state space over these chains. It is simple to show that the state space of $\mathcal{A}$ is connected. We note that it is a standard result that all perfect matchings in a complete bipartite graph are connected via edge swaps [33]. Moreover, the space of pseudographs can be seen exactly as the set of all perfect matchings over the disconnected complete bipartite degree neighborhoods in the joint degree matrix configuration model. The connectivity result is much less obvious for $\mathcal{B}$. We adapt a result of Taylor [33] that all graphs with a given degree sequence are connected via edge swaps in order to prove this. The proof is inductive and follows the structure of Taylor's proof. It is included in the Appendix.

THEOREM 3.1. *Given two simple graphs, G_1 and G_2 of the same size with the same joint degree matrix, there exists a series of endpoint rewirings to transform G_1 into G_2 (and vice versa) where every intermediate graph is also simple.*

4 Mixing Time of the Markov Chain

The Markov chain $\mathcal{A}$ is very similar to one analyzed by Kannan, Tetali and Vempala [12]. We can exactly use their canonical paths and analysis to show that the mixing time is polynomial. This result follows directly from Theorem 3 of [12] for chain $\mathcal{A}$. This is because the joint degree matrix configuration model can be viewed as $|\mathcal{D}|$ complete, bipartite, and disjoint components. These components should remain disjoint, so the Markov Chain can be viewed as a 'meta-chain' which samples a component and then runs one step of the Kannan, Tetali and Vempala chain on that component. Even though the mixing time for this chain is provably polynomial, this upper bound is too large to be useful in practice.

The analysis to bound the mixing time for $\mathcal{B}$ chain is significantly more complicated. We considered using the canonical path method to bound the congestion of this chain. The standard trick is to define a path from G_1 to G_2 that fixes the misplaced edges identified by $G_1 \oplus G_2$ in a globally ordered way. However, this is difficult to do for chain $\mathcal{B}$ because fixing a specific edge may not be atomic, i.e. from the proof of Theorem 3.1 it may take up to 4 swaps to correctly connect a vertex with an endpoint if there are conflicts with the other degree neighborhoods. These swaps take place in other degree neighborhoods and are not local moves. In addition, step (4) also prevents us from using path coupling as a proof of the mixing time.

Given that bounding the mixing time of this chain seems to be difficult, we use a series of experiments that substitute the *autocorrelation time* for the mixing time.

4.1 Autocorrelation Time

Autocorrelation time is a quantity that is related to the mixing time and is popular among physicists. We will give a brief introduction to this concept, and refer the reader to Sokal's lecture notes for further details [31].

The autocorrelation of a signal is the cross-correlation of the signal with itself given a lag t. More formally, given a series of data $\langle X_i \rangle$ where each X_i is a drawn from the same distribution X with mean μ and variance σ, the autocorrelation function is $R_X(t) = \frac{E[(X_i - \mu)(X_{i-t} - \mu)]}{\sigma^2}$.

DEFINITION 3. *The exponential autocorrelation time is $\tau_{exp,X} = \limsup_{t \to \infty} \frac{t}{-\log |R_X(t)|}$ [31].*

DEFINITION 4. *The integrated autocorrelation time is $\tau_{int,X} = \frac{1}{2} \sum_{t=-\infty}^{\infty} R_X(t) = \frac{1}{2} + \sum_{t=1}^{\infty} R_X(t)$ [31].*

Intuitively, an inherent problem with a Markov Chain method is that successive states generated by the chain may be highly correlated. If we were able to draw independent samples from the stationary distribution, then the autocorrelation of that set of samples with itself would go 0 as the number of samples increased.

The autocorrelation time is capturing the size of the gaps between sampled states of the chain needed before the autocorrelation of this 'thinned' chain is very small. If the thinned chain has 0 autocorrelation, then it must be exactly sampled from the stationary distribution. In practice, when estimating the autocorrelation from a finite number of samples, we do not expect it to go to exactly 0, but we do expect it to 'die away' as the number of samples and gap increases.

The difference between the exponential autocorrelation time and the integrated autocorrelation time is that the exponential autocorrelation time measures the time it takes for the chain to reach equilibrium after a cold start, or 'burn-in' time. The integrated autocorrelation time is related to the increase in the variance over the samples from the Markov Chain as opposed to samples that are truly independent. Often, these measurements are the same, although this is not necessarily true.

We can substitute the autocorrelation time for the mixing time because they are, in effect, measuring the same thing - the number of iterations that the Markov Chain needs to run for before the difference between the current distribution and the stationary distribution is small. We will use the integrated autocorrelation time estimate.

4.2 Experimental Design We used the Markov Chain $\mathcal{B}$ in two different ways. First, for each of the datasets, we ran the chain for 50,000 iterations 15 times. We used this to calculate the the autocorrelation values for each edge for each lag between 100 and 15,000 in multiples of 100. From this, we calculated the estimated integrated autocorrelation time, as well as the iteration time for the autocorrelation of each edge to drop under a threshold of 0.001. This is discussed in Section 4.3.

We also replicated the experimental design of Raftery and Lewis [28]. Given our estimates of the autocorrelation time for each size graph in Section 4.3, we ran the chain again for long enough to capture 10,000 samples where each sample had x iterations of the chain between them. x was chosen to vary from much smaller than the estimated autocorrelation time, to much larger. From these samples, we calculated the sample mean for each edge, and compared it with the actual mean from the joint degree matrix. We looked at the total variational distance between the sample means and actual means and showed that the difference appears to be converging to 0. We chose the mean as an evaluation metric because we were able to calculate the true means theoretically. We are unaware of another similarly simple metric.

We used the formulas for empirical evaluation of mixing time from page 14 of Sokal's survey [31]. In particular, we used the following:

- The sample mean is $\overline{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- The sample unnormalized autocorrelation function is $\hat{C}(t) = \frac{1}{n-t}\sum_{i=1}^{n-t}(x_i - \overline{\mu})(x_{i+t} - \overline{\mu})$.
- The natural estimator of $R_X(t)$ is $\hat{\rho}(t) = \hat{C}(t)/\hat{C}(0)$
- The estimator for $\tau_{int,X}$ is $\hat{\tau}_{int} = \frac{1}{2}\sum_{t=-(n-1)}^{n-1}\lambda(t)\hat{\rho}(t)$ where λ is a 'suitable' cutoff function.

Data Sets We have used several publicly available datasets, Word Adjacencies [27], Les Miserables [15], American College Football [8], the Karate Club [34], and the Dolphin Social Network [20]. In the following $|V|$ is the number of nodes, $|E|$ is the number of edges and $|\mathcal{J}|$ is the number of non-zero entries in the joint degree matrix.

	$\|V\|$	$\|E\|$	$\|\mathcal{J}\|$
AdjNoun	112	425	159
Dolphins	62	159	61
Football	115	616	18
Karate	34	78	40
LesMis	77	254	99

We selected these datasets because of their size. For a sequence of length x, calculating the autocorrelation of gap t requires $(x-t)^2$ dot products. Our experiments require that we calculate the autocorrelation for each possible edge in a graph for many lags. Thus running the full set of experiments requires $O(|V|^2 x \log x)$ time and is prohibitive when V is large. In Section 4.6 we discuss results that suggest a more feasible method for estimating autocorrelation time for larger graphs.

4.3 Autocorrelation Values For each dataset and each run we calculated the unnormalized autocorrelation values for each edge for t between 100 and 15,000 in multiples of 100. We randomly selected 1 run for each dataset and graphed the autocorrelation values for each of the edges. We present the data for the Karate and Dolphins datasets in Figures 4 and 5 while the graphs for the other datasets are included in the Appendix due to their similarity to the two presented.

All of the graphs exhibit the same behavior. We see an exponential drop off initially, and then the autocorrelation values oscillate around 0. This behavior is due to the limited number of samples, and a bias due to using the sample mean for each edge. If we ignore the noisy tail, then we estimate that the autocorrelation

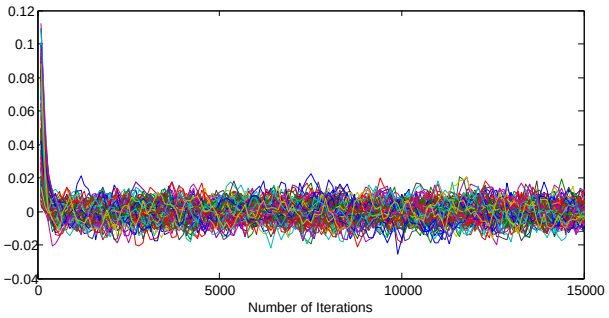

Figure 4: The exponential dropoff for Karate appears to end after 400 iterations.

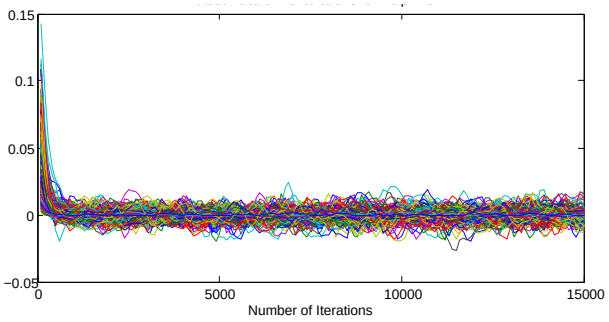

Figure 5: The exponential dropoff for Dolphins appears to end after 600 iterations.

'dies off' at the point where the mean absolute value of the autocorrelation approximately converges, then we can locate the 'elbow' in the graphs in Table 1.

4.4 Estimated Integrated Autocorrelation Time For each dataset and run, we calculated the estimated integrated autocorrelation time. Given that we calculated the autocorrelation in lags of 100 from 100 to 15,000 for each dataset, we estimate $\hat{\rho}(t)$ as 100 times the sum of the values. The cut-off function we used was $\lambda(t) = 1$ if $0 < t < 15,000$ and 0 otherwise. This value was calculated for each edge.

For each dataset, we calculated the following: the mean, median and maximum values for the estimated integrated autocorrelation time for each edge. These are graphed in Figure 6. The number of the edges represents each of the datasets. In particular, in order from left to right they are Karate, Dolphins, LesMis, AdjNoun and Football. The error bars represent the maximum and minimum values over all edges, while the line runs through the median value over all edges.

All three metrics give roughly the same picture. We

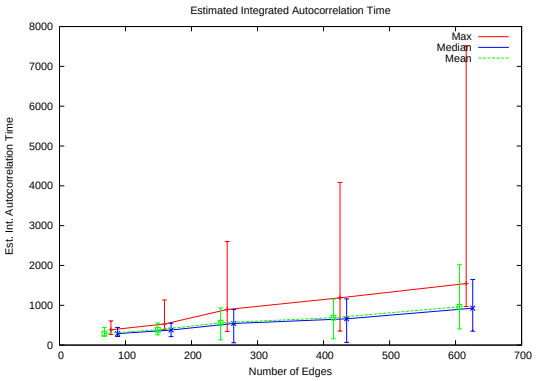

Figure 6: The max, median and min values over the edges for the est. integrated autocorrelation times. L to R: Karate, Dolphins, LesMis, Adjnoun and Football

note that there is much higher variance in estimated autocorrelation time for the larger graphs. If we consider just the median values, then the autocorrelation time appears to be linear. However, if we consider the error bars for the maximum then we may need a superlinear time to guarantee convergence of all edges.

4.5 The Sample Mean Approaches the Real Mean for Each Edge Given the results of the previous experiment estimating the integrated autocorrelation time, we next executed an experiment suggested by Raftery and Lewis [28]. First we note that for each edge e, we know the true value of $P(e \in G|G \text{ has } \mathcal{J})$ is exactly $\frac{\mathcal{J}_{k,l}}{\mathcal{D}_k \mathcal{D}_l}$ or $\frac{\mathcal{J}_{k,k}}{\binom{\mathcal{D}_k}{2}}$ if e is an edge between degrees k and l. This is because there are $\mathcal{D}_k \mathcal{D}_l$ potential (k,l) edges that show up in any graph with a fixed $\mathcal{J}$, and each graph has $\mathcal{J}_{k,l}$ of them. If we consider the graphs as being labeled, then we can see that each edge has an equal probability of showing up when we consider permutations of the orderings.

Thus, our experiment was to take samples at varying intervals, and consider how the sample mean of each edge compared with our known theoretical mean. For all graphs, we took 10,000 samples at varying gaps depending on our estimated integrated autocorrelation time. For the smaller graphs, we took 10 different samples of 10,000 edges. We elided this step in the larger graphs because we saw very small variance. Additionally, we saw that the total variational distance quickly converged to a small, but non-zero value. For the smaller graphs, Karate and Dolphins, we repeated the experiment with 20,000 samples and show that this error is due to the

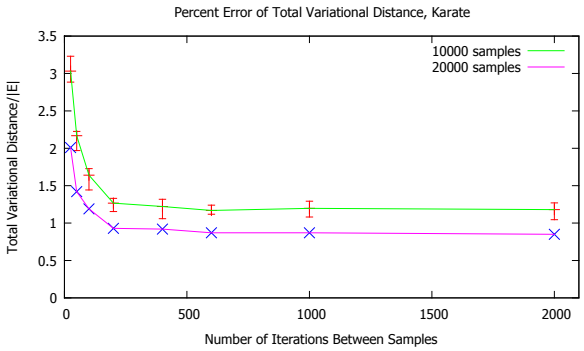

Figure 7: The total variational distance as a percentage of the sum of μ_e. This was repeated for 10 runs with 10,000 samples each and 1 run with 20,000 samples. The error bars represent the maximum and minimum error over the 10 runs. The line is the median values.

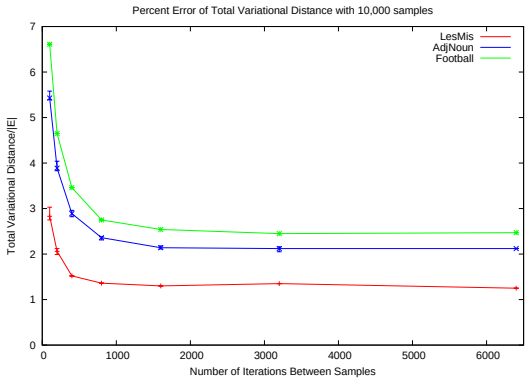

Figure 8: The total variational distance as a percentage of the sum of μ_e. This is 1 run with 10,000 samples for AdjNoun, Football and LesMis

number of samples and not the sampler. We present these results in Figure 9. If $S_{e,g}$ is the sample mean for edge e and gap g, and μ_e is the true mean, then the graphed value is $\sum_e |S_{e,g} - \mu_e| / \sum_e \mu_e$.

In all of the figures, the line runs through the median error for the 10 runs and the error bars are the maximum and minimum values. We note that the maximum and minimum are very close to the median as they are within 0.05% for most intervals. These graphs imply that we are sampling uniformly after a gap of 200 for the Karate graph. For the dolphin graph, we see very similar results, and note that the error becomes constant after a sampling gap of 400 iterations.

For the larger graphs, we took just one series of samples for each of the following gaps: 100, 200, 400, 800, 1600, 3200, and 6400. Again, we see consistent results, although the residual error is higher. This is to be expected because there are more potential edges in these graphs, so we took relatively fewer samples per edge. For AdjNoun, we appear to be sampling uniformly between a gap of 800 and 1600. For Football, the error converges between 800 and 1600 again. LesMis also appears to have converge between the 800 and 1600 gaps. These results are slightly better than the median estimated integrated autocorrelation time for each of the datasets.

4.6 Relationship Between Mean of an Edge and Autocorrelation From the results in Section 4.3, we considered if there was a relationship between the time it took for the autocorrelation of an edge e to 'die down' and μ_e. For each edge and each run, we calculated the first time where $\hat{\rho}_e(t)$ passed under a threshold (0.001). From these values, we looked at the mean time to pass

	$\lvert E\rvert$	Max EI	Mean Conv.	Thresh.
AdjNoun	425	1186	800-1600	700
Dolphins	159	528	400-600	600
Football	616	1546	800-1600	900
Karate	78	382	200-400	400
LesMis	254	894	800-1600	1000

Table 1: A summary of estimates on convergence from the three experiments. The values are the Maximum Estimated Integrated Autocorrelation time (Max EI), the Sample Mean Convergence iteration number, and the time to drop under the Autocorrelation Threshold. The Autocorrelation threshold was calculated as when the average absolute value of the autocorrelation was less than 0.0001

under the threshold and created Figures 10, 11, and 12. We have included the graphs for Football and Dolphins in the Appendix because they have a smaller range of ratios and illustrate the effect less well.

From these graphs, we suspect that there is a relationship between μ_e and the time to pass under a threshold. Unfortunately, none of our datasets contained a significant number of edges with larger μ_e values, i.e. between 0.5 and 1. In order to test this hypothesis, we designed a synthetic dataset that contained the many edges with values of μ_e at $\frac{i}{20}$ for $i = 1, \cdots 20$. We describe the creation of this dataset in the appendix.

The final dataset we created had 326 edges, 194 vertices and 21 distinct $\mathcal{J}$ entries. We ran the Markov Chain 200 times for this synthetic graph. For each run, we calculated the threshold value for each edge.

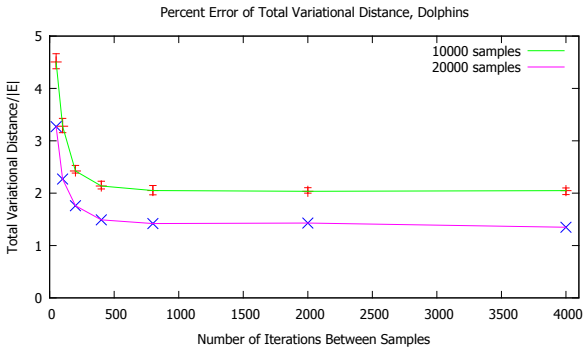

Figure 9: The total variational distance as a percentage of the sum of μ_e. This was repeated for 10 runs with 10,000 samples each and 1 run with 20,000 samples. The error bars represent the maximum and minimum error over the 10 runs. The line is the median values.

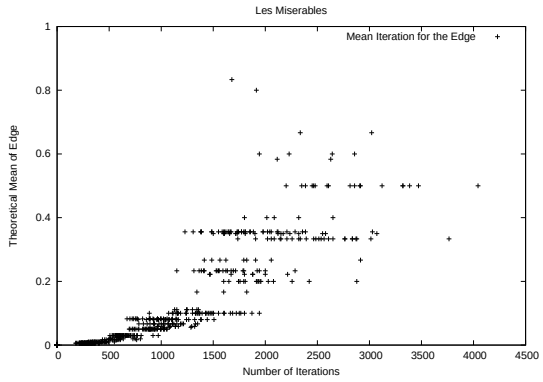

Figure 10: The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge for LesMis.

Figure 13 shows the edges' mean vs its mean time for the autocorrelation value to pass under 0.001. We see that there is a roughly symmetric curve that obtains its maximum at $\mu_e = 0.5$.

This result suggests a way to estimate the autocorrelation time for larger graphs without repeating the entire experiment for every edge that could possibly appear. One could calculate μ_e for each edge and sample edges with $0.4 \leq \mu_e \leq 0.6$. One could then repeat our experiments for just these selected edges in order to estimate the autocorrelation time.

5 Conclusions and Future Work

This paper makes two primary contributions. The first is the investigation of Markov Chain methods for uniformly sampling graphs with a fixed joint degree distribution. Previous work shows that the mixing time of $\mathcal{A}$ is polynomial, while our experiments suggest that the mixing time of $\mathcal{B}$ is also polynomial. The relationship between the mean of an edge and the autocorrelation values can be used to efficiently experiment with larger graphs by sampling edges with mean between 0.4 and 0.6 and repeating the analysis for just those edges. This would allow one to efficiently obtain estimates of the running time for much larger graphs. Initial experimental results for larger graphs following this design show a similar polynomial running time.

Our second contribution is in the design of the experiments to evaluate the mixing time of the Markov Chain. In practice, it seems the stopping time for sampling is often chosen without justification. Autocorrelation is a simple metric to use, and can be strong evidence that a chain is close to the stationary distribution when used correctly.

Acknowledgments The authors would like to acknowledge helpful contributions of David Gleich, Satish Rao, Jaideep Ray, Alistair Sinclair, Virginia Vassilevska Williams and Wes Weimer.

References

[1] W. Aiello, F. R. K. Chung, and L. Lu. A random graph model for massive graphs. *STOC*, pages 171–180, 2000.

[2] Y. Amanatidis, B. Green, and M. Mihail. Graphic realizations of joint-degree matrices. *Manuscript*, 2008.

[3] M. Bayati, J. H. Kim, and A. Saberi. A sequential algorithm for generating random graphs. *APPROX-RANDOM*, pages 326–340, 2007.

[4] J. Blitzstein and P. Diaconis. A sequential importance sampling algorithm for generating random graphs with prescribed degrees, 2006.

[5] A. Z. Broder. How hard is it to marry at random? (on the approximation of the permanent). *STOC*, pages 50–58, 1986.

[6] M. Faloutsos, P. Faloutsos, and C. Faloutsos. On power-law relationships of the internet topology. *SIGCOMM*, pages 251–262, 1999.

[7] A. Flaxman, A. Frieze, and J. Vera. A geometric preferential attachment model of networks. *WAW*, pages 44–55, 2004.

[8] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proc. Natl. Acad. Sci. USA*, 99:7821–7826, 2002.

[9] C. Gkantsidis, M. Mihail, and E. W. Zegura. The markov chain simulation method for generating connected power law random graphs. *ALENEX*, pages 16–25, 2003.

[10] M. Jerrum and A. Sinclair. Fast uniform generation of regular graphs. *Theor. Comput. Sci.*, 73(1):91–100, 1990.

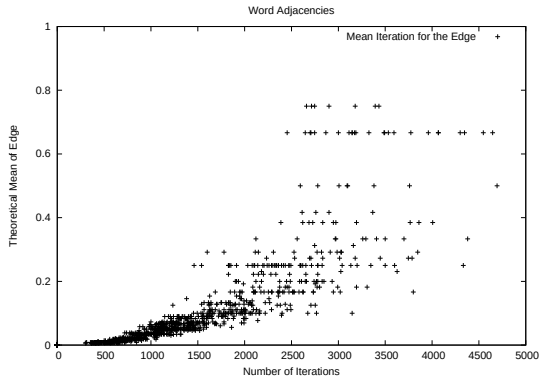

Figure 11: The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge for AdjNoun.

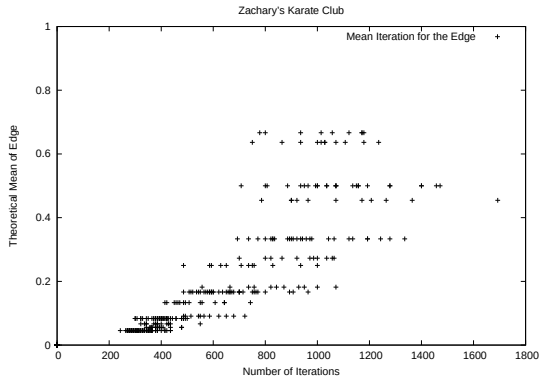

Figure 12: The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge for Karate.

[11] M. Jerrum, A. Sinclair, and E. Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *J. ACM*, 51(4):671–697, 2004.

[12] R. Kannan, P. Tetali, and S. Vempala. Simple markov-chain algorithms for generating bipartite graphs and tournaments. *Random Struct. Algorithms*, 14(4):293–308, 1999.

[13] J. H. Kim and V. Vu. Generating random regular graphs. *Combinatorica*, 26(6):683–708, 2006.

[14] J. Kleinberg. Small-world phenomena and the dynamics of information. *NIPS*, pages 431–438, 2001.

[15] D. E. Knuth. The stanford graphbase: A platform for combinatorial computing, 1993.

[16] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Random graph models for the web graph. *FOCS*, pages 57–65, 2000.

[17] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research (JMLR)*, 11:985–1042, 2010.

[18] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. *KDD*, pages 177–187, 2005.

[19] J. Leskovec, K. Lang, A. Dasgupta, and M. Mahoney. Statistical properties of community structure in large social and information networks. *WWW*, pages 695–704, 2008.

[20] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, 2003.

[21] P. Mahadevan, C. Hubble, D. V. Krioukov, B. Huffaker, and A. Vahdat. Orbis: rescaling degree correlations to generate annotated internet topologies. *SIGCOMM*, pages 325–336, 2007.

[22] P. Mahadevan, D. Krioukov, K. Fall, and A. Vahdat. Systematic topology analysis and generation using degree correlations. *SIGCOMM*, pages 135–146, 2006.

[23] M. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70(5):056131, Nov 2004.

[24] M. Newman. Modularity and community structure in networks. *PNAS*, 103:8577–82, 2006.

[25] M. E. J. Newman. Assortative mixing in networks. *Phys. Rev. Letter*, 89:208701, May 20 2002.

[26] M. E. J. Newman. Mixing patterns in networks. *Phys. Rev. E.*, 67:026126, February 04 2002.

[27] M. E. J. Newman. Finding community structure in networks using the eigenvectors of matrices, 036104. *Phys. Rev. E*, 74, 2006.

[28] A. E. Raftery and S. M. Lewis. The number of iterations, convergence diagnostics and generic metropolis algorithms. *Practical Markov Chain Monte Carlo*, pages 115–130, 1995.

[29] E. Ravasz and A. L Barabasi. Hierarchical organization in complex networks. *Phys. Rev. E*, 67:026112, 2003.

[30] A. Sinclair and M. Jerrum. Approximate counting, uniform generation and rapidly mixing markov chains. *Inf. Comput.*, 82(1):93–133, 1989.

[31] A. Sokal. Monte carlo methods in statistical mechanics: Foundations and new algorithms, 1996.

[32] A. Steger and N. C. Wormald. Generating random regular graphs quickly. *Combinatorics, Probability & Computing*, 8(4), 1999.

[33] R. Taylor. Constrained switching in graphs. *SIAM J. ALG DISC. METH.*, 3, 1:115–121, 1982.

[34] W. W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33:452–473, 1977.

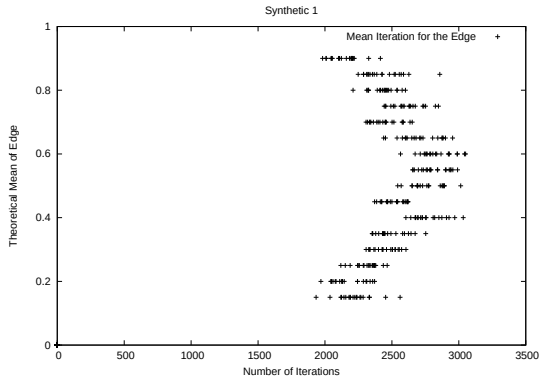

Figure 13: The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge. This synthetic dataset has a larger range of μ_e values than the real datasets and a significant number of edges for each value.

6 Appendix

Proof of Theorem 2.3: Let $\mathcal{J}$ be given and $\mathcal{D}$ be the associated degree sequence. As with the Erdős-Gallai condition, let $d_1 \geq d_2 \geq \cdots d_n$ be the sorted degree sequence. We assume only that $\mathcal{J}_{k,l} \leq \mathcal{D}_k\mathcal{D}_l$ for $k \neq l$ and $\mathcal{J}_{k,k} \leq \binom{\mathcal{D}_k}{2}$. For clarity later, double each $\mathcal{J}_{k,k}$ entry so that $k\mathcal{D}_k = \sum_l \mathcal{J}_{k,l}$ instead of $k\mathcal{D}_k = \mathcal{J}_{k,k} + \sum_l \mathcal{J}_{k,l}$.

We want to show that $\sum_{i=1}^{k} d_i \leq k(k-1) + \sum_{i=k+1}^{n} \min\{k, d_i\}$ for every k. For clarity, we first present the argument when $d_k > d_{k+1}$. Also, let $d_k = l$.

$$\sum_{i=1}^{k} d_i = \sum_{x=l}^{n}\sum_{y=1}^{n} \mathcal{J}_{x,y} = \sum_{x=l}^{n}\sum_{y=l}^{n} \mathcal{J}_{x,y} + \sum_{x=l}^{n}\sum_{y=1}^{l-1} \mathcal{J}_{x,y}$$

First, we note that $\sum_{x=l}^{n}\sum_{y=l}^{n} \mathcal{J}_{x,y} \leq \sum_{x=l}^{n}\sum_{y=l}^{n} \mathcal{D}_x\mathcal{D}_y = \sum_{x=l}^{n} \mathcal{D}_x \sum_{y=l}^{n} \mathcal{D}_y \leq k^2$. However, we wanted to show it was less than $k(k-1)$. This is true because for k $\mathcal{J}$ values, it's true that $\mathcal{J}_{x,x} \leq \mathcal{D}_x(\mathcal{D}_x - 1)$. Intuitively, the sum is including a self-loop for every node that can't possibly exist.

Now, we consider $\sum_{x=l}^{n}\sum_{y=1}^{l-1} \mathcal{J}_{x,y}$. Here, let us fix y and note that it contributes $\sum_{x=l}^{n} \mathcal{J}_{x,y}$. This is at most $y\mathcal{D}_y$ on one hand, and also at most $\sum_{x=l}^{n} \mathcal{D}_y\mathcal{D}_x = \mathcal{D}_y \sum_{x=l}^{n} \mathcal{D}_x \leq \mathcal{D}_y k$ on the other. Therefore, $\sum_{x=l}^{n} \mathcal{J}_{x,y} \leq \min\{y\mathcal{D}_y, \mathcal{D}_y k\} = \mathcal{D}_y \min\{y, k\}$. This is exactly the quantity we desired, so $\sum_{x=l}^{n}\sum_{y=1}^{l-1} \mathcal{J}_{x,y} \leq \sum_{i=k+1}^{n} \min\{k, d_i\}$.

We now address the case where $d_k = d_{k+1}$. If we let $l = d_k$ again, then the above argument changes because $\sum_{i=1}^{k} d_k = \sum_{x=l}^{n}\sum_{y=1}^{n} \mathcal{J}_{x,y} - (\mathcal{D}_l - z)l$ where $d_{k-z}, \cdots d_k = l$. We note that the restricted graphical conditions here are that when we consider the edges with at least one endpoint in $\{d_1, \cdots d_k\}$, we have that $\mathcal{J}_{x,l} \leq z\mathcal{D}_x$ (and $z(z-1)$ where appropriate). Plugging this into the above argument results in exactly the right values, as before.

Proof of Lemma 3.1: Let C_1 and C_2 be two neighboring configurations in $\mathcal{B}$. This means that they differ by exactly 4 edges in exactly 1 degree neighborhood. Let this degree be k and let these edges be e_1v_1 and e_2v_2 in C_1 whereas they are e_1v_2 and e_2v_1 in C_2. We want to show that C_1 and C_2 have exactly the same number of feasible k-degree swaps.

Without loss of generality, let e_x, e_y be a swap that is prevented by e_1 in C_1 but allowed in C_2. This must mean that e_x neighbors v_1 and e_y neighbors some $v_y \neq v_1, v_2$. Notice that the swap e_1e_x is currently feasible. However, in C_2, it is now infeasible to swap e_1, e_x, even though e_x and e_y are now possible.

If we consider the other cases, like e_x, e_y is prevented by both e_1 and e_2, then after swapping e_1 and e_2, e_x, e_y is still infeasible. If swapping e_1 and e_2 makes something feasible in C_1 infeasible in C_2, then we can use the above argument in reverse. This means that the number of feasible swaps in a k-neighborhood is invariant under k-degree swaps.

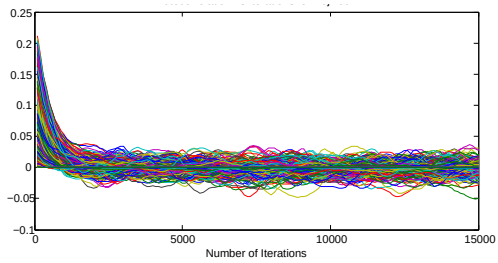

Figure 14: The exponential dropoff for the AdjNoun data appears to end after 700 iterations.

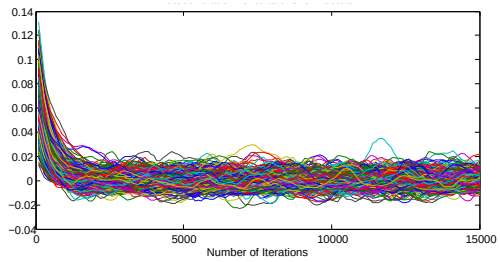

Figure 15: The exponential dropoff for the Football data appears to end after 900 iterations.

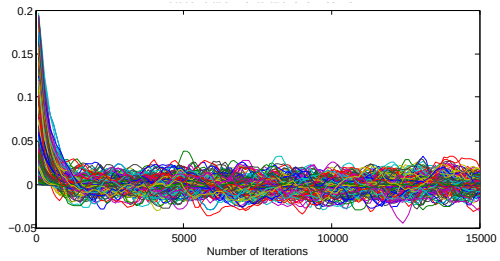

Figure 16: The exponential dropoff for the LesMis data appears to end after 1000 iterations.

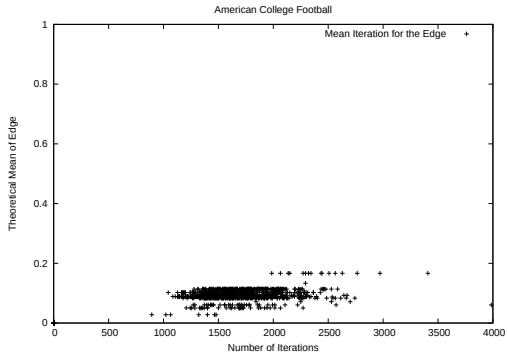

Figure 17: The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge.

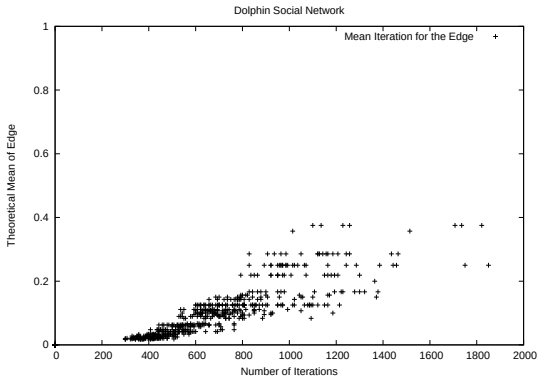

Figure 18: The time for an edge's estimated autocorrelation function to pass under the threshold of 0.001 versus μ_e for that edge.